Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations

Daniel S. Brown *1 Wonjoon Goo *1 Prabhat Nagarajan 2 Scott Niekum 1

Abstract

A critical flaw of existing inverse reinforcement learning (IRL) methods is their inability to significantly outperform the demonstrator. This is a consequence of the general reliance of IRL algorithms upon some form of mimicry, such as feature-count matching, rather than inferring the underlying intentions of the demonstrator that may have been poorly executed in practice. In this paper, we introduce a novel reward learning from observation algorithm, Trajectory-ranked Reward EXtrapolation (T-REX), that extrapolates beyond a set of (approximately) ranked demonstrations in order to infer high-quality reward functions from a set of potentially poor demonstrations. When combined with deep reinforcement learning, we show that this approach can achieve performance that is more than an order of magnitude better than the best-performing demonstration, as well as state-of-the-art imitation learning and IRL methods, on multiple Atari and MuJoCo benchmark tasks. Finally, we demonstrate that T-REX is robust to modest amounts of ranking noise, opening up future possibilities for automating the ranking process, for example, by watching a learner noisily improve at a task over time.

1. Introduction

Due to advantages such as computational speed, precise manipulation, and exact timing, computers and robots are often better than humans at performing tasks with well-defined goals and objectives. However, it can be difficult, even for experts, to design reward functions and objectives that lead to desired behaviors when designing autonomous agents (Ng et al., 1999; Amodei et al., 2016). When goals or rewards are difficult for a human to specify, inverse reinforcement learning (IRL) (Abbeel & Ng, 2004) techniques can be applied to infer the goals of a user from demonstrations. Unfortunately, high-quality demonstrations are often difficult to provide for many tasks—for instance, consider a non-expert user attempting to give kinesthetic demonstrations of a household chore to a robot. Even for relative experts, tasks such as high-frequency stock trading or playing complex video games can be difficult to perform optimally.

If a demonstrator is suboptimal, but his intentions can be ascertained, then a learning agent ought to be able to exceed the demonstrator’s performance in principle. However, current IRL algorithms fail to do this, typically searching for a reward function that makes the demonstrations appear near-optimal (Abbeel & Ng, 2004; Ramachandran & Amir, 2007; Ziebart et al., 2008; Wulfmeier et al., 2015; Finn et al., 2016; Henderson et al., 2018). Thus, when the demonstrator is suboptimal, IRL results in suboptimal behavior as well. Other imitation learning approaches (Argall et al., 2009) that mimic behavior directly without reward inference, such as behavioral cloning (Torabi et al., 2018a), also suffer from the same problem.

To overcome this critical flaw in current imitation learning methods, we propose a novel IRL algorithm, Trajectory-ranked Reward EXtrapolation (T-REX) that utilizes a ranking amongst the demonstrations to extrapolate a user’s underlying intent beyond the best demonstration, even when
all demonstrations are highly suboptimal. This, in turn, enables a reinforcement learning agent to exceed the performance of the demonstrator by learning to optimize this extrapolated reward function. Specifically, we use (approximately) ranked demonstrations to learn a state-based reward function that assigns greater total return to higher-ranked trajectories. By learning a reward function that is a function of state only, we are additionally able to learn from observations alone without action labels.

Utilizing ranking in this way has several advantages. First, rather than imitating suboptimal demonstrations, it allows us to identify the trend of which features are better and worse, in a manner that can be extrapolated beyond the demonstrations. Although the learned reward function could potentially overfit to the provided rankings, we demonstrate empirically that it extrapolates well, successfully predicting returns of trajectories that are significantly better than any observed demonstration, likely due to the powerful regularizing effect of having many pairwise ranking constraints between trajectories. For example, the degenerate all-zero reward function (the agent always receives a reward of 0) makes any given set of demonstrations appear optimal, but is eliminated from consideration by any pair of (non-equally) ranked demonstrations. Second, when learning features directly from high-dimensional data, this regularizing effect can also help to prevent overfitting to the small fraction of state space visited by the demonstrator. By utilizing a set of suboptimal, but ranked demonstrations, we provide the neural network with diverse data from multiple areas of state space, and, unlike learning from expert demonstrations, rankings among suboptimal demonstrations allow an agent to learn what not to do.

We evaluate T-REX on a variety of standard Atari and MuJoCo benchmark tasks. Our experiments show that T-REX can extrapolate well, achieving performance that is sometimes more than an order of magnitude better than the best-performing demonstration, as well as outperforming state-of-the-art imitation learning algorithms. Finally, we show that T-REX performs well even in the presence of significant ranking noise, and provide results showing that T-REX can learn simply good policies simply by observing a novice demonstrator that noisily improves at a challenging task over time.

2. Related Work

The goal of our work is to achieve improvements over a sub-optimal demonstrator in high-dimensional reinforcement learning tasks without requiring a hand-specified reward function, active feedback, or supervision during policy learning. While there is a large body of research on learning from demonstrations (Argall et al., 2009; Gao et al., 2012; Osa et al., 2018; Arora & Doshi, 2018), most work assumes access to action labels, while we learn only from observations. Additionally, little work has addressed the problem of learning from ranked demonstrations, especially when they are significantly suboptimal. To the best of our knowledge, our work is the first to show better-than-demonstrator performance in high-dimensional tasks such as Atari, without requiring active human supervision or access to ground truth rewards.

2.1. Learning from demonstrations

Early work on learning from demonstration focused on behavioral cloning (Pomerleau, 1991), in which the goal is to learn a policy that imitates the actions taken by the demonstrator; however, without substantial human feedback and correction, this method is known to have large generalization error (Ross et al., 2011). Recent deep learning approaches to imitation learning (Ho & Ermon, 2016) have used Generative Adversarial Networks (Goodfellow et al., 2014) to model the distribution of actions taken by the demonstrator.

Rather than directly learn to mimic the demonstrator, inverse reinforcement learning (IRL) (Gao et al., 2012; Arora & Doshi, 2018) seeks to find a reward function that models the intention of the demonstrator, thereby allowing generalization to states that were unvisited during demonstration. Given such a reward function, reinforcement learning (Sutton & Barto, 1998) techniques can be applied to learn an optimal policy. Much work on IRL focuses on matching the expected feature counts of an expert policy (Ng & Russell, 2000; Abbeel & Ng, 2004). Maximum entropy IRL does this matching, while further disambiguating inference by also maximizing the entropy of the resulting policy (Ziebart et al., 2008; Boularias et al., 2011; Wulfmeier et al., 2015; Finn et al., 2016). While matching expected feature counts enables the learned policy to achieve the same reward as the demonstrator, it can be problematic if the demonstrator follows a suboptimal policy.

Syed & Schapire (2008) proved that, given prior knowledge about which features contribute positively or negatively to the true reward, an apprenticeship policy can be found that is guaranteed to outperform the demonstrator. However, their approach requires hand-crafted, linear features, knowledge of the true signs of the rewards features, and also requires repeatedly solving an MDP in the inner loop of the algorithm. Our proposed method uses deep learning and ranked demonstrations to automatically learn complex features that are positively and negatively correlated with performance, and is able to outperforms the demonstrator by only solving a single RL problem.

Our work can be considered a form of preference-based IRL (PBIRL) (Wirth et al., 2016; Sugiyma et al., 2012) which is an extension the IRL framework that considers a set of demonstrations with preference labels. However,
most existing approaches only consider reward functions that are linear in known features and have not studied extrapolation capabilities. For a more complete survey of preference based reinforcement learning, see (Wirth et al., 2017). Other methods (Burchfiel et al., 2016; El Asri et al., 2016) have proposed to use quantitatively scored trajectories as opposed to qualitative pairwise preferences over demonstrations. However, none of these aforementioned methods have been applied to the types of high-dimensional deep reinforcement learning tasks considered in this paper.

2.2. Learning from observation

Recently there has been a shift towards learning from observations, in which the actions taken by the demonstrator are unknown. Torabi et al. (2018a) propose a state-of-the-art model-based approach to perform behavioral cloning from observation. Sermanet et al. (2018) and Liu et al. (2018) propose methods to learn directly from a large corpora of videos containing multiple view points of the same task. Yu et al. (2018) propose a meta-learning approach that can learn to perform a task from a single demonstration, but requires training on a wide variety of different, but related tasks in order to learn a strong reward prior. Henderson et al. (2018) and Torabi et al. (2018b) extend Generative Adversarial Imitation Learning (Ho & Ermon, 2016) to remove the need for action labels, but both are based on Generative Adversarial Networks (Goodfellow et al., 2014) which are difficult to train and have been shown to fail to scale to high-dimensional imitation learning tasks such as Atari (Tucker et al., 2018).

2.3. Learning from suboptimal demonstrations

Very little work has tried to learn good policies from suboptimal demonstrations. Grollman & Billard (2011) propose a method that learns from failed demonstrations where a human attempts, but is unable, to perform a task; however, for this method to work, demonstrations must be labeled as failures and manually clustered into two sets of demonstrations: those that overshoort and those that undershoot the goal. Shiarlis et al. (2016) demonstrate that if successful and failed demonstrations are labeled and the reward function is a linear combination of known features, then maximum entropy IRL can be used to find a policy whose expected feature counts match those of the successful demonstrations while not matching those of the failed demonstrations. Zheng et al. (2014) and Choi et al. (2019) propose methods that are robust to small numbers of unlabeled suboptimal demonstrations, but require a majority of expert demonstrations in order to learn which demonstrations are anomalous.

In reinforcement learning, it is common to initialize a policy from suboptimal demonstrations and then improve this policy using the ground truth reward signal (Kober & Peters, 2009; Taylor et al., 2011; Hester et al., 2017; Gao et al., 2018). However, designing appropriate rewards for reinforcement learning can be difficult for non-experts and can easily lead to unintended behaviors (Ng et al., 1999; Amodei et al., 2016).

2.4. Reward learning for video games

Most deep learning-based methods for reward learning require access to demonstrator actions and do not scale to high-dimensional tasks such as video games (e.g. Atari) (Ho & Ermon, 2016; Finn et al., 2016; Fu et al., 2017; Qureshi & Yip, 2018). Tucker et al. (2018) extend state-of-the-art Adversarial Inverse Reinforcement Learning (Fu et al., 2017) to the Atari domain, but are only able to achieve a maximum performance that is four times worse than the expert demonstrator.

Aytar et al. (2018) use video demonstrations of experts to learn good policies for Montezuma’s Revenge, Pitfall, and Private Eye. However, they accomplish this through the use of extremely high-performing demonstrations which their method is unable to outperform. Furthermore, their method is inherently different from ours in that their learned reward function is designed to imitate the demonstration gameplay, rather than extrapolate beyond the capabilities of the demonstrator.

Christiano et al. (2017) develop an algorithm that learns to play Atari games via actively-collected pairwise preferences over trajectories during policy learning. This approach requires obtaining thousands of labels through constant human supervision during policy learning. In contrast, our method only requires an initial set of (approximately) ranked set of demonstrations as input and learns a better-than-demonstrator policy without requiring human supervision during policy learning. Ibarz et al. (2018) combine deep Q-learning from demonstrations (Hester et al., 2017) and active preference learning (Christiano et al., 2017) to learn to play Atari games. However, this approach often results in performance that is significantly worse than the demonstrator and requires both initial demonstrations as well as thousands of active queries during policy learning. Additionally, Ibarz et al. require demonstrator actions: demonstrator actions are required to initialize Q-values using an action-based, large-margin loss (Hester et al., 2017) and (s, a, s’) transitions from the demonstrations need to be contained in the Q-learning replay buffer.

To the best of our knowledge, our work is the first to significantly outperform a demonstrator without using ground truth rewards or active preference queries during policy learning. Our approach also does not require demonstrator actions and is able to infer the demonstrator’s intention without requiring any environmental interactions.
3. Problem Definition

We model the environment as a Markov decision process (MDP) consisting of a set of states $S$, actions $A$, transition probabilities $P$, reward function $r$, and discount factor $\gamma$ (Puterman, 2014). A policy, $\pi$, is a mapping from states to probabilities over actions, $\pi(a|s) \in [0,1]$. Given a policy and an MDP, the expected discounted return of the policy is given by $J(\pi) = \mathbb{E}(\sum_{t=0}^{\infty} \gamma^t r_t | \pi)$.

In this work we are concerned with the problem of inverse reinforcement learning from observation, where we do not have access to the reward function of the MDP. An agent is a distribution of demonstrations $\mathcal{D}$ consisting of trajectories (sequences of states), and seeks to recover the reward function that the demonstrator is attempting to optimize. Given a sequence of $m$ ranked trajectories $\tau_t$ for $t = 1, \ldots, m$, where $\tau_i \prec \tau_j$ if $i < j$, we wish to find a parameterized reward function, $\hat{r}_\theta$, that approximates the true reward function $r$. Given $\hat{r}_\theta$, we then seek to optimize a policy $\hat{\pi}$ that can outperform the demonstrations. Furthermore, many imitation learning methods learn from sequences of state-action pairs, we consider trajectories that only consist of states, making our method problem one of learning from observation (Liu et al., 2018; Sermanet et al., 2018; Torabi et al., 2018a).

We only assume access to a qualitative ranking over demonstrations. Thus, we only require the demonstrator to have an internal goal or intrinsic reward. The demonstrator can rank trajectories using any method, such as giving pairwise preferences over demonstrations or by rate each demonstration on a scale. Note that even if the relative scores of the demonstrations are used to form a qualitative ranking, it is still necessary to infer why some trajectories are better than others, which is what we propose to do.

4. Methodology

We now describe Trajectory-ranked Reward EXtrapolation (T-REX), our proposed algorithm for using ranked suboptimal demonstrations to extrapolate a user’s underlying intent beyond the best demonstration. Given a sequence of $m$ demonstrations ranked from worst to best, $\tau_1, \ldots, \tau_m$, T-REX has two steps: (1) reward inference and (2) policy optimization.

Given the ranked demonstrations, T-REX performs reward inference by approximating the reward at state $s$ using a neural network, $\hat{r}_\theta(s)$, such that $\sum_{s \in \tau_i} r_\theta(s) < \sum_{s \in \tau_j} r_\theta(s)$ when $\tau_i \prec \tau_j$. The parameterized reward function $\hat{r}_\theta$ can be trained with ranked demonstrations using the generalized loss function:

$$L(\theta) = \mathbb{E}_{\tau_i, \tau_j \sim \Pi} \left[ \xi \left( \mathbb{P}(\hat{J}_\theta(\tau_i) < \hat{J}_\theta(\tau_j)), \tau_i \prec \tau_j \right) \right], \quad (1)$$

where $\Pi$ is a distribution of demonstrations, $\xi$ is a binary classification loss function, $\hat{J}$ is a (discounted) return defined by a parameterized reward function $r_\theta$, and $\prec$ is an indication of the preference between the demonstrated trajectories.

We specifically represent the probability $\mathbb{P}$ as a softmax-normalized distribution and we instantiate $\xi$ using a cross-entropy loss:

$$L(\theta) \approx -\sum_{\tau_i, \tau_j} \log \frac{\exp \sum_{s \in \tau_i} \hat{r}_\theta(s)}{\exp \sum_{s \in \tau_i} \hat{r}_\theta(s) + \exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}, \quad (2)$$

This loss function trains a classifier that can predict whether one trajectory is preferable to another based on the predicted returns of each trajectory. This form of loss function follows from the classic Bradley-Terry and Luce-Shephard models of preferences (Bradley & Terry, 1952; Luce, 2012) and has shown success when learning preferences from active queries (Christiano et al., 2017; Ibarz et al., 2018).

To increase the number of training examples, T-REX trains on partial trajectory pairs rather than full trajectory pairs. In particular, T-REX makes the simplifying assumption that any partial trajectory from a higher ranked trajectory is preferred to any partial trajectory of the same length from a lower ranked trajectory. This results in noisy preference labels that are only weakly supervised; however, using partial trajectories allows T-REX to learn expressive neural network reward functions from only a small number of ranked demonstrations. During training we randomly select pairs of trajectories, $i$ and $j$. We then randomly select partial trajectories $\tilde{\tau}_i$ and $\tilde{\tau}_j$ of length $L$. For each observation in each partial trajectory we perform a forward pass through the network $\hat{r}_\theta$ and sum the predicted rewards for each observation to compute the cumulative return for each partial trajectory. We then use the predicted cumulative returns as the logit values in the cross-entropy loss with the label corresponding to the higher ranked demonstration.

Given the learned reward function $\hat{r}_\theta(s)$, T-REX then seeks to optimize a policy $\hat{\pi}$ with better-than-demonstrator performance through reinforcement learning using $\hat{r}_\theta$.

5. Experiments and Results

5.1. Mujoco

We first evaluated our proposed method on three robotic locomotion tasks using the Mujoco simulator (Todorov et al.,
Extrapolating Beyond Suboptimal Demonstrations

2012) within OpenAI Gym (Brockman et al., 2016), namely HalfCheetah, Hopper, and Ant. In all three tasks, the goal of the robot agent is to move forward as fast as possible without falling to the ground.

5.1.1. Demonstrations

To generate demonstrations, we trained a PPO (Schulman et al., 2017) agent with the ground-truth reward for 500 training steps (64,000 simulation steps) and saved its policy after every 5 training steps. This provides us with different policies of varying quality. To generate demonstrations, we divided the checkpointed policies into different overlapping stages based on ground-truth returns. We used 3 stages for HalfCheetah and Hopper. For HalfCheetah we used the first 9, 12, and 24 trajectories, respectively. For Hopper we used the first 9, 12, and 18 trajectories. For Ant we used two stages consisting of the first 12 and 40 trajectories, respectively. We used the PPO implementation from OpenAI Baselines (Dhariwal et al., 2017) with the given default hyperparameters.

5.1.2. Experimental Setup

For each checkpoint within a stage, we generated a trajectory of length 1,000 and ranked the trajectories using the ground-truth return (accumulated sum of rewards). We trained the reward network using 5,000 random pairs of partial trajectories of length 50, with preference labels based on the trajectory rankings, not the ground-truth return of the partial trajectories. To prevent overfitting, we represented the reward function using an ensemble of five deep neural networks, trained separately with different random pairs. Each network has 3 fully connected layers of 256 units with ReLU nonlinearities. We train the reward network using the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 1e-4 and a minibatch size of 64 for 10,000 time steps.

To evaluate the quality of our learned reward, we then trained a policy to maximize the inferred reward function via PPO. The outputs of each of the five reward networks in our ensemble, \( r(s) \), are normalized by their standard deviation in order to compensate for any scale differences amongst the models. The reinforcement learning agent receives the average of the ensemble as the reward, plus the control penalty used in OpenAI Gym (Brockman et al., 2016). This control penalty represents a standard safety prior over reward functions for robotics tasks, namely to minimize joint torques. We found that optimizing a policy based solely on this control penalty does not lead to forward locomotion, thus learning a reward function from demonstrations is still necessary.

5.1.3. Results

Learned Policy Performance We measured the performance of the policy learned by T-REX under the ground-truth reward function. We compared against Behavior Cloning from Observations (BCO) (Torabi et al., 2018a), a state-of-the-art learning from observation method. BCO trains a policy via supervised learning, which has been shown to be competitive with state-of-the-art IRL (Ho & Ermon, 2016) on MuJoCo tasks without requiring action labels (Torabi et al., 2018a), making it one of the best baselines when learning from observations. We trained BCO using only the best demonstration among the available suboptimal demonstrations (see Appendix A for full BCO implementation details).

We compared against three different levels of suboptimality (Stage 1, 2, and 3), corresponding to increasingly better demonstrations. The results are shown in Table 1. The policies learned by T-REX perform significantly better than the provided suboptimal trajectories in all the stages of HalfCheetah and Hopper. This provides evidence that T-REX can discover reward functions that extrapolate beyond the performance of the demonstrator, while behavioral cloning fails to perform better than the average of the demonstrations across all tasks.

Reward Extrapolation We next investigated the ability of T-REX to accurately extrapolate beyond the demonstrator. To do so, we compared ground-truth return and T-REX-inferred return across trajectories from a range of performance qualities, including trajectories much better than the best demonstration given to T-REX. The extrapolation of the reward function learned by T-REX is shown in Figure 2. The plots in Figure 2 give insight into the performance of T-REX. When T-REX learns a reward function that has a strong positive correlation between the ground-truth reward function and the inferred reward function, then it is able to surpass the performance of the suboptimal demonstrations. However, in Ant the correlation is not as strong, resulting in worse-than-demonstrator performance in Stage 2.

We visualized the T-REX-learned policy for HalfCheetah in Figure 3. Visualizing the demonstrations from different stages shows the specific way the policy evolves over time; an agent learns to crawl first and starts to try to walk in an up-straight position. The T-REX policy learned from the highly suboptimal Stage 1 demonstrations results in a similar style crawling gait; however, T-REX captures some of the intention behind the demonstration and is able to optimize a gait that resembles the demonstrator but with increased speed, resulting in a better-than-demonstrator policy. Similarly, given demonstrations from Stage 2, which are still highly suboptimal, T-REX learns a policy that resembles the gait of the best demonstration, but is able to optimize and partially stabilize this gait. Finally, given demonstrations
Table 1: The results on three robotic locomotion tasks when given suboptimal demonstrations. For each stage and task, the best performance given suboptimal demonstrations is shown on the top row, and the best achievable performance (i.e. performance achieved by a PPO agent) under the ground-truth reward is shown at the bottom. The best performance using our method and the behavior cloning baseline are displayed at the middle. The mean and standard deviation are based on 25 trials (obtained by running PPO five times and for each run of PPO performing five policy rollouts).

<table>
<thead>
<tr>
<th>Task</th>
<th>HalfCheetah</th>
<th>Hopper</th>
<th>Ant</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stage 1</td>
<td>Stage 2</td>
<td>Stage 3</td>
</tr>
<tr>
<td>Demo Performance</td>
<td>12.52 (1.04)</td>
<td>44.98 (0.60)</td>
<td>89.87 (8.15)</td>
</tr>
<tr>
<td>T-Rex (ours)</td>
<td>46.90 (1.89)</td>
<td>61.56 (10.96)</td>
<td>143.40 (3.84)</td>
</tr>
<tr>
<td>T-REX (time-ordered)</td>
<td>51.39 (4.52)</td>
<td>54.90 (2.29)</td>
<td>154.67 (57.43)</td>
</tr>
<tr>
<td>BCO</td>
<td>7.71 (8.35)</td>
<td>23.59 (8.33)</td>
<td>57.13 (19.14)</td>
</tr>
<tr>
<td>GAIL</td>
<td>7.39 (4.12)</td>
<td>8.42 (3.43)</td>
<td>26.28 (12.73)</td>
</tr>
<tr>
<td>Best w/</td>
<td>199.11 (9.08)</td>
<td>15.94 (1.47)</td>
<td>182.23 (8.98)</td>
</tr>
<tr>
<td>GT Reward</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

from Stage 3, which are still suboptimal, T-REX is able to learn a near-optimal gait.

5.2. Atari

In the next set of experiments, we evaluated T-REX on the eight Atari games shown in Table 2.

5.2.1. Demonstrations

For each game we generated 12 full-episode trajectories using PPO policies checkpointed every 50 training updates for all games except for Seaquest where we used every 5th training update due to the faster learning in this game.

5.2.2. Experimental Setup

We used an architecture for reward learning similar to the one proposed in (Ibarz et al., 2018), with four convolutional layers with sizes 7x7, 5x5, 3x3, and 3x3, with strides 3, 2, 1, and 1. Each convolutional layer used 16 filters and LeakyReLU non-linearities. We then used a fully connected layer with 64 hidden units and a single scalar output. We fed in stacks of 4 frames with values normalized between 0 and 1 and set the value of the upper 10 rows of pixels to zero in order to mask the ground-truth Atari scores. We optimized the reward function using Adam with a learning rate of 1e-4. Given the learned reward function, we optimized a policy by training a PPO agent on the learned reward function for 40 million frames. We normalized the learned reward function by feeding the output of \( r_\theta(s) \) through a sigmoid function before passing it to the PPO algorithm. After learning a reward, we train a PPO agent on the learned reward function for 40 million frames to obtain our learned policy.

5.2.3. Results

**Learned Policy Performance**  The average performance of T-REX under the ground-truth reward function and the best and average performance of the demonstrator are shown in Table 2. We also compare against Behavioral Cloning from Observation (BCO) (Torabi et al., 2018a) (see Appendix A for details). We compare against BCO, since current state-of-the-art IRL methods have been shown to not yet scale to Atari (Tucker et al., 2018), and because BCO has been shown to be competitive with state-of-the-art IRL techniques such as GAIL on simpler domains (Ho & Ermon, 2016). We also ran experiments using Generative Adversarial Imitation Learning from Observations (BCO) (Torabi et al., 2018b); however, the method completely failed when applied to Atari games, even when given many expert demonstrations. Table 2 shows that T-REX outperforms BCO in 7 out of 8 games. More importantly, we are able to outperform the average performance of the demonstrator in 6 out of 8 games and outperform the best demonstration in 5 out of 8 games. In comparison, BCO performs worse than the average performance of the demonstrator in all games.

In Table 3 we also compare against the approach proposed by Ibarz et al. (2018) which combines Deep Q-learning from demonstrations and active preference queries (DQfD+ΔA) on seven Atari games (Space Invaders results are not available for DQfD+ΔA). DQfD+ΔA uses demonstrations consisting of \( (s_t, a_t, s_{t+1}) \)-tuples and use these demonstrations to initialize a policy using DQfD (Hester et al., 2017). The algorithm
then uses the active preference learning algorithm of (Christiano et al., 2017) to refine the inferred reward function and initial policy learned from demonstrations. The first two columns of Table 3 compare the demonstration quality given to DQfD+A and T-REX. While our results make use of more demonstrations (12 for T-REX versus 4–7 for DQfD+A), our demonstrations are typically orders of magnitude worse than the demonstrations used by DQfD+A: on average the demonstrations given to DQfD+A are 38 times better than those used by T-REX. However, despite this large gap in the performance of the demonstrations, T-REX surpasses the performance of DQfD+A on Q*Bert, and Seaquest. We achieve these results using 12 ranked demonstrations. Assuming worst-case complexity $O(n^2)$ for sorting this requires only 144 comparisons by the demonstrator. In comparison, the DQfD+A results used 3,400 preference labels obtained during policy training using ground-truth rewards.

We also examine the ability of prior work to exceed the performance of the demonstrator. In Table 3 we have marked with * results that surpass the best demonstration. DQfD+A only surpass the demonstrator in 3 out of 9 games, even with thousands of active queries. Note that DQfD+A extends the original DQfD algorithm Hester et al. (2017), which uses demonstrations combined with RL on ground-truth rewards, yet only able to surpass the best demonstration in 14 out of 41 Atari games. In contrast, we are able to leverage only 12 ranked demos to achieve better-than-demonstrator performance on 5 out of 8 games, without requiring access to true rewards or access to thousands of active queries from an oracle.

**Reward Extrapolation** We also examined the extrapolation of the reward function learned using T-REX. Results are shown in Figure 4. We see that accurate extrapolation between normalized predicted and ground truth returns are achieved in Beam Rider, Seaquest, and Space Invaders—
Table 3. Best demonstrations and average performance of learned policies for T-REX (ours) and the DQfD with active preference learning (DQfD+A) (Ibarz et al., 2018). Results for T-REX are averaged over 25 trials and 5 seeds. Results that exceed the best demonstration are marked with an asterisk (*). Note that T-REX requires at most only 121 preference labels (for ranking 12 demonstrations), whereas the DQfD+A results use 3.4k labels queried during policy learning. DQfD+A also use action labels on the demonstrations, whereas T-REX learns purely from observation.

<table>
<thead>
<tr>
<th>Game</th>
<th>Ranked Demos</th>
<th>LfO Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best</td>
<td>Avg.</td>
</tr>
<tr>
<td>Beam Rider</td>
<td>1188</td>
<td>802.3</td>
</tr>
<tr>
<td>Breakout</td>
<td>33</td>
<td>15.1</td>
</tr>
<tr>
<td>Enduro</td>
<td>84</td>
<td>40</td>
</tr>
<tr>
<td>Hero</td>
<td>13235</td>
<td>6742</td>
</tr>
<tr>
<td>Pong</td>
<td>-6</td>
<td>-15.6</td>
</tr>
<tr>
<td>Q*bert</td>
<td>800</td>
<td>627</td>
</tr>
<tr>
<td>Seaquest</td>
<td>600</td>
<td>373.3</td>
</tr>
<tr>
<td>Space Invaders</td>
<td>600</td>
<td>332.9</td>
</tr>
</tbody>
</table>

Table 4. Best demonstrations and average performance of learned policies for T-REX (ours) and the DQfD with active preference learning (DQfD+A) (Ibarz et al., 2018). Results for T-REX are averaged over 25 trials and 5 seeds. Results that exceed the best demonstration are marked with an asterisk (*). Note that T-REX requires at most only 121 preference labels (for ranking 12 demonstrations), whereas the DQfD+A results use 3.4k labels queried during policy learning. DQfD+A also use action labels on the demonstrations, whereas T-REX learns purely from observation.

<table>
<thead>
<tr>
<th>Game</th>
<th>Ranked Demos</th>
<th>LfO Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg.</td>
<td>Best</td>
</tr>
<tr>
<td>Space Invaders</td>
<td>246.7</td>
<td>385</td>
</tr>
<tr>
<td>Q*bert</td>
<td>439.6</td>
<td>875</td>
</tr>
<tr>
<td>Ms Pacman</td>
<td>378.3</td>
<td>630</td>
</tr>
</tbody>
</table>

5.4. Robustness to noisy demonstrations

All experiments thus far have had access to ground-truth rankings. We now explore the effects of noise in the ranking process on the performance of T-REX in the stage 1 Hopper task. We synthetically generated ranking noise by randomly swapping adjacent trajectories in a sorted trajectory list over several passes. By varying the number of swaps, we were able to generate a different noise levels. The noise level is measured as a total order correctness; the fraction of all pairwise comparisons with the correct rank-ordering.

The results of this experiment, averaged over 25 runs in each noise setting, are shown in Figure 5. It shows that our algorithm is relatively unaffected by noise up to around 15% noise, after which performance plummets. This robustness to moderate noise suggests that it may be possible to learn without any hand-specified rankings at all, by simply observing a novice demonstrator improve at a challenging task (noisily) over time, using time as a surrogate for ranking. It also implies that T-REX can likely be used successfully with noisy human labelers.

6. Conclusion and Future Work

In this paper, we introduced T-REX, a reward learning technique for high-dimensional, deep reinforcement learning tasks that can learn to extrapolate intent from suboptimal ranked demonstrations. To the best of our knowledge, this is the first IRL algorithm that is able to significantly outperform the demonstrator without additional external knowledge (e.g., signs of feature contributions to reward) and that scales to high-dimensional Atari games. When combined with deep reinforcement learning, we showed that this approach can achieve performance that is more than an order of magnitude better than the best-performing demonstration, as well as a state-of-the-art behavioral cloning from observation method, on multiple Atari and MuJoCo benchmark.
Extrapolating Beyond Suboptimal Demonstrations

(a) Beam Rider  (b) Breakout  (c) Enduro  (d) Hero

(e) Pong  (f) Q*bert  (g) Seaquest  (h) Space Invaders

Figure 4. Extrapolation plots for Atari games. We compare ground truth returns over demonstrations to the predicted returns using T-REX (normalized to be in the same range as the ground truth returns). The black solid line represents the performance range of the demonstrator. The green dashed line represents extrapolation.

Figure 5. The performance of T-REX for different amounts of ranking noise in the Hopper domain. T-REX shows robust performance as long as less than 15% of the pairwise preference labels induced by the ranked demonstrations are incorrect. The reward function is trained on stage 1 Hopper demonstrations, and results are averaged across 25 trials.

...tasks. Finally, we demonstrated that T-REX is robust to modest amounts of ranking noise, opening up future possibilities for automating the ranking process, for example, by watching a learner noisily improve at a task over time.

References


Extrapolating Beyond Suboptimal Demonstrations


Extrapolating Beyond Suboptimal Demonstrations


A. Behavioral Cloning from Observation

TODO: put details here.